ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.11, November 2010, pp.2701−2710
doi: 10.3724/SP.J.1001.2010.03660

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

逆向工程中的大类图拆分方法*

刘 辉[1,2,4+], 邵维忠[2,3], 麻志毅[2,3]

[1](北京理工大学 计算机学院,北京 100081)

[2](高可信软件技术教育部重点实验室,北京 100871)

[3](北京大学 信息科学技术学院 软件研究所,北京 100871)

[4](北京理工大学 计算机学院 智能信息技术北京市重点实验室, 北京 100081)

Decomposition of Large Class Diagrams Generated by Reverse Engineering

LIU Hui[1,2,4+], SHAO Wei-Zhong[2,3], MA Zhi-Yi[2,3]

[1](School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China)

[2](Key Laboratory of High Confidence Software Technologies, Ministry of Education, Beijing 100871, China)

[3](Software Institute, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

[4](Beijing Laboratory of Intelligent Information Technology, School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China)

+ Corresponding author: E-mail: liuhui04@sei.pku.edu.cn

Liu H, Shao WZ, Ma ZY. Decomposition of large class diagrams generated by reverse engineering. *Journal of Software*, 2010,21(11):2701−2710. http://www.jos.org.cn/1000-9825/3660.htm

Abstract: This paper proposes an approach to decompose large class diagrams. It first collects metrics of coupling among classifiers (classes and interfaces). According to the principle of high cohering and low coupling, it breaks low coupling classifiers while showing high coupling classifiers in the same diagrams. To guarantee that the generated new class diagrams are readable, it confines sizes of new diagrams to a predefined scope. The results of its evaluations on industrial projects suggest that the approach is practical and valuable. The approach proposed in this paper helps to improve the readability of software models.

Key words: class diagram; unified modeling language (UML); reverse engineering; software maintenance

摘 要: 提出了一种大类图拆分方法.首先通过度量工具计算类图中类目(类及接口)间的耦合度.根据面向对象设计中高内聚低耦合的设计原则,将紧耦合的类目划入同一个类图,而耦合度低的类目间实现分离.为了确保生成的类图大小合适,拆分方法对每个类图的大小进行限定,将每个类图的大小限定在预先定义的合理区间内.

* Supported by the National Natural Science Foundation of China under Grant Nos.60773152, 61003065 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2007AA010301 (国家高技术研究发展计划(863)); the Open Research Fund Program of Key Laboratory of High Confidence Software Technologies, Ministry of Education, China, under Grant No.HCST200802 (高可信软件技术教育部重点实验室开放研究基金); the Open Research Fund Program of Beijing Laboratory of Intelligent Information Technology (智能信息技术北京市重点实验室开放研究基金); the Basic Research Found of Beijing Institute of Technology of China (北京理工大学基础研究基金)

Received 2008-10-23; Accepted 2009-06-01

通过在实际系统中的应用,拆分方法的合理性和有效性得到验证.该大类图拆分方法有利于逆向工程的进一步完善,有利于提高软件模型的可读性和可理解性.

关键词: 类图;UML;逆向工程;软件维护

中图法分类号: TP311 **文献标识码:** A

软件模型是软件开发与维护中的核心制品.与于源代码相比,软件模型具有较高的抽象层次、较好的可读性和可理解性.因此,软件模型在软件开发和维护中具有不可替代的作用.尤其是在软件维护中,软件维护人员可能没有参与软件的开发,需要通过软件本身(包括代码、文档和模型等)来理解待维护的软件系统.阅读软件模型是维护人员理解软件系统的主要途径之一.

但是,许多软件系统并没有提供软件模型.其原因之一是软件开发过程不规范,没有严格的分析与设计阶段,所以最后提交的软件制品没有系统的分析与设计模型.另一个可能的原因是商业模式.开发团队虽然设计了详细的软件模型,但出于商业原因没有给软件维护人员提供软件的分析与设计模型.无论出于何种原因,缺少软件模型都将使软件维护更加困难.

解决缺乏分析设计模型的主要方法之一是通过逆向工程(reverse engineering)[1]从代码生成软件模型.逆向工程是一项年轻的、快速发展中的技术.现有的主流建模工具都提供了一定的逆向工程能力.Rational Rose[2], Together[3],JUDE[4],JBOO[5]等都可以从源代码中生成以统一建模语言(unified modeling language,简称 UML)[6]表示的软件模型.目前,逆向工程得到的模型多半限于系统的静态模型(某些逆向工程工具也可以生成诸如顺序图等动态模型,但目前生成的动态模型的实用价值还有待进一步提升)[7].类图(class diagram)是逆向工程中最常见的,也是最有用的模型之一.

邵维忠和杨芙清[8]认为,在面向对象模型中,类图是最基本的、最重要的模型,而其他各种图(比如顺序图、状态图和用况图)则起辅助作用(具体分析详见该书第 3 章).构成类图的主要成分包括类、属性、操作以及类之间的关系(继承、聚合、关联、依赖等).类图中的类、属性和操作都与源代码中的相应概念一一对应,所以类图的逆向生成较为直接.这也是类图在逆向工程中获得较好支持的一个重要原因.

虽然类图在逆向工程中获得了较好的支持,但是目前主流的逆向工具所生成的类图往往过于庞大,影响类图的阅读和理解.如图 1 所示的类图即为 JUDE 在元建模平台 Meta-Modeler[9]上逆向工程得到的一个类图(经过手工调整和优化布局).这个类图包含的元素(类及接口)多达 35 个.按照“7±2”原则[10],人在一段时间内能够理解和记忆的事物是有一定数量限制的,一般不超过 7 ± 2, 所以,多达 35 个元素的庞大类图不利于软件维护人员的阅读和理解.在类图的显示上也会遇到问题,比如图 1 所示的类图就很难看清其中的文字(类名、属性、操作等).如果要放大类图以看清文字,又难以完整地展示类图的结构,从而难以看清元素之间的关系.所以,我们需要某种方法将逆向工程生成的大类图拆分为多个大小合适的小类图,以提高类图的可读性和可理解性.

大类图的生成与现有逆向工具的工作方式有关.现有的逆向工具通常按类(接口)的命名空间(namespace)来组织,将同一个命名空间下的所有类和接口都放入同一个类图.另外一些工具则根据包(package)或目录(directory)来组织,将某个目录及子目录下的所有类及接口组织为一个类图.但在实际工程中,一个命名空间或者一个目录下可能包含大量的类及接口定义.还有部分逆向工程工具甚至将项目所包含的所有类目都放入一个单一的类图中.这种超大类图的可读性极差.

为了解决逆向工程生成的类图难以阅读和理解的问题,本文提出了一种将大类图拆分为多个大小合适的紧凑类图的方法.将紧耦合的类目划分到同一个类图,而松耦合的类目之间实现分离,从而控制类图的大小.

本文第 1 节介绍大类图的拆分方法.第 2 节通过实验验证本文提出的拆分方法的实用性.第 3 节对拆分方法进行讨论.第 4 节介绍并讨论相关工作.第 5 节总结本文工作并指出进一步的研究方向.

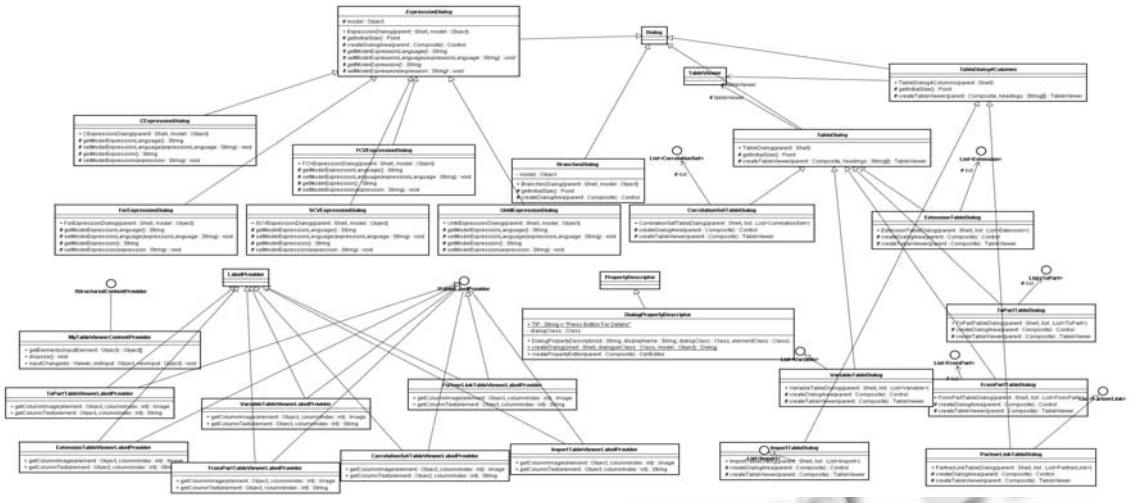

Fig.1 Class diagram generated by JUDE

图 1 JUDE 逆向工程生成的类图

1 类图拆分方法

本节首先介绍类目(classifier,包括类和接口等[6])之间的耦合关系.基于类目间的耦合关系,本文提出了一种拆分大类图的方法.

1.1 逆向工程生成的类图

类图是展现系统静态结构的主要视图.主要元素包括类目(类和接口)以及类目间的关系(关联、依赖、继承、组合、聚合).类可以包含属性和操作,接口也可以包含操作.因为本文的类图拆分算法并不拆分类目的内部结构,所以将类和接口都作为单一的元素来考虑.聚合关系表示类目之间的整体-部分关系.而组合关系是聚合的一个特例,表示部分和整体具有相同的生命周期或者部分只为该整体所拥有.目前,逆向工程工具通常不区分这两种关系,统一表示为聚合关系,所以本文也将聚合和组合统一表示为聚合.基于以上分析,可以将逆向工程生成的类图表示为

$$ClassDiagram=\{C,Ass,Dep,Inh,Agg\} \tag{1}$$

其中,C 表示类图包含的类目;Ass 表示类图中的关联关系,如果类目 $c_1,c_2\in C$ 且有从 c_1 到 c_2 的关联关系,则 $\langle c_1,c_2\rangle\in Ass$,如果是双向关联,则 $\langle c_1,c_2\rangle\in Ass\wedge\langle c_2,c_1\rangle\in Ass$;与 Ass 类似,Dep 表示类图中的依赖关系,Inh 表示类图中的继承关系,Agg 表示类图中的聚合关系.

1.2 耦 合

耦合是软件度量中的一个重要概念.Stevens 等首次提出耦合的概念,并将耦合定义为“模块间关联强度的度量”.为了度量面向对象系统的耦合关系,人们已经提出了大量的度量指标及度量框架[11],比如对象间耦合(coupling between objects,简称 CBO)、消息传递耦合(message passing coupling,简称 MPC)、数据抽象耦合(data abstraction coupling,简称 DAC)等.

CBO 的定义为“类的 CBO 是与该类具有耦合关系的类的个数”[12,13].如果某个类使用了其他类的方法或属性,则具它们之间具有耦合关系.CBO 的大小可以度量某个类与周围环境的关联度.CBO 越大,该类与周围环境的关系就越紧密.

CBO 的形式定义为

$$CBO(c)=\{d\in C-\{c\}\mid uses(c,d)\vee uses(d,c)\} \quad (2)$$

其中,C 表示系统的所有类的集合;$uses(c,d)$表示类 c 使用了类 d 的属性或方法.CBO 一般从源代码中分析统计,而不必从生成的类图中计算.

类的 MPC 是该类中定义的向其他类发送消息的语句(send statements)的数量[14].MPC 用于表征某个类的实现对其他类的方法的依赖程度.

这些度量指标都是以类为单位,度量某个类与周围环境的相互依赖程度.而在类图的拆分中,我们主要关心的是每对类目之间的耦合强度.而且类图作为一种表现系统静态结构的特殊视图,主要展现的是类间的继承、聚合(组合)、关联、依赖等关系,而系统动态行为不是类图所要展现的主要信息.考虑类图中各种关系的受关注程度,类目间的关系(继承、聚合、关联和依赖)在划分类图中的影响力从高到低依次为继承、聚合(组合)、关联、依赖[8].为便于类图的拆分,定义类目间耦合如下:

$$Coupling(c_1,c_2)=\{IsAncestor(c_2,c_1)\times 8+Aggregated(c_1,c_2)\times 4+Assciated(c_1,c_2)\times 2+Depend(c_1,c_2)\} \quad (3)$$

如果 $\langle c_2,c_1\rangle\in Inh$, 则 $IsAncestor(c_2,c_1)$ 为 1, 否则, $IsAncestor(c_1,c_2)$ 为 0; 类似地, 如果 $\langle c_2,c_1\rangle\in Agg$, 则 $Aggregated(c_1,c_2)$ 为 1,否则为 0;如果 $\langle c_2,c\rangle\in Ass$, 则 $Associated(c_1,c_2)$ 为 1,否则为 0;如果 $\langle c_2,c_1\rangle\in Dep$, 则 $Depend(c_1,c_2)$ 为 1,否则为 0.

这里将继承、聚合、关联和依赖对类图拆分的影响因子进行量化,其因子依次递减.以 8,4,2,1 作为因子可以保证有继承关系的类目之间比没有继承关系的类目之间具有更强的耦合度.与此类似,后续 3 个影响因子(聚合、关联和依赖)的量化也可以保证相应的优先顺序.

1.3 类图拆分算法

拆分算法基于第 1.2 节的耦合关系,将强耦合的类放入同一个类图,同时分离耦合度较低的类.这种拆分方法符合面向对象设计中的高内聚、低耦合的设计原则.

拆分算法的流程如图 2 所示.其中,迭代部分的详细流程描述如图 3 所示,类图中的每个类目均需要一次迭代过程.最后零散类目的处理流程如图 4 所示.

具体的算法过程详细解释如下:

首先计算每对类目之间的耦合强度.现有的软件度量工具可以自动收集并统计类及接口之间的耦合强度.同时统计每个类及接口的 CBO 以便作为后续迭代的排序依据.

按照 CBO 从高到低对待拆分类图中的类和接口进行排序.CBO 度量某个类与周围环境的耦合广度.CBO 越高,表示有越多的类目与该类具有耦合关系,所以,以该类为起点,根据耦合度选出合适子类图的可能性就越大.

将类图中的所有类目的“已分配”标志位标记为“*false*”.“已分配”标志位用于指明类目是否已经出现在某个新创建的小类图中.在开始拆分大类图之前,所有类目的标记位都应该为“*false*”.

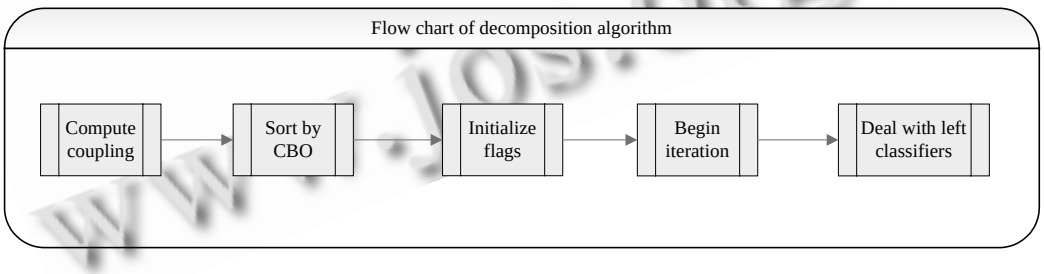

Fig.2 Flow chart of decomposition algorithm

图 2 拆分算法流程示意图

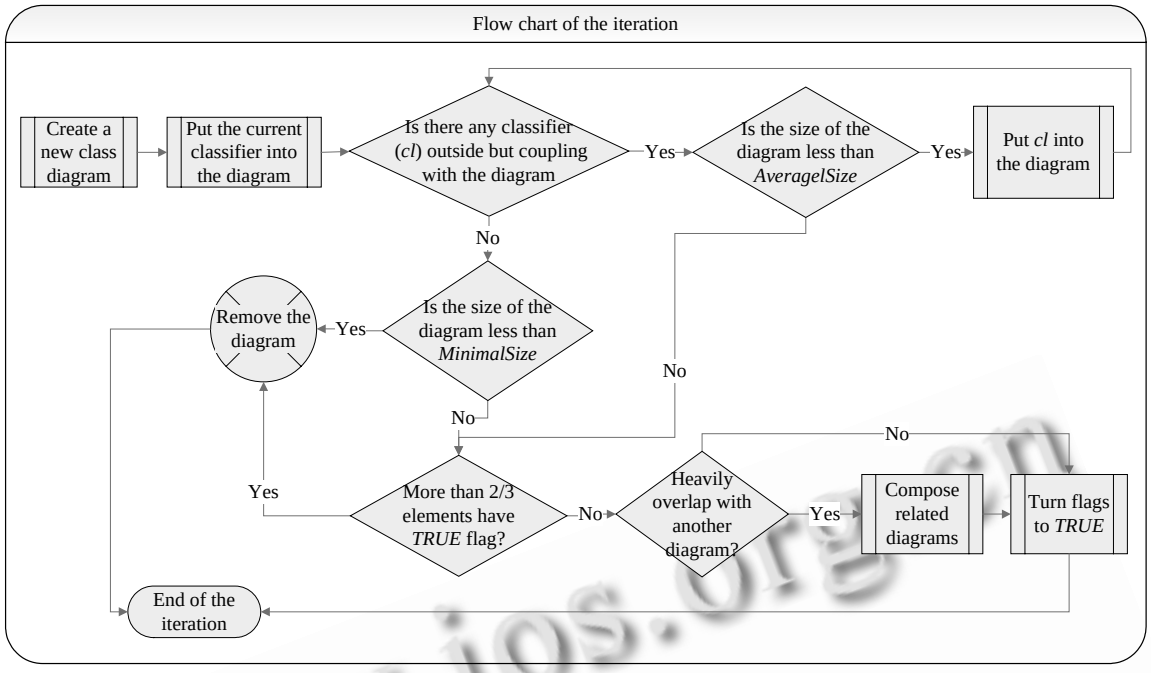

Fig.3 Flow chart of the iteration

图 3 迭代处理流程示意图

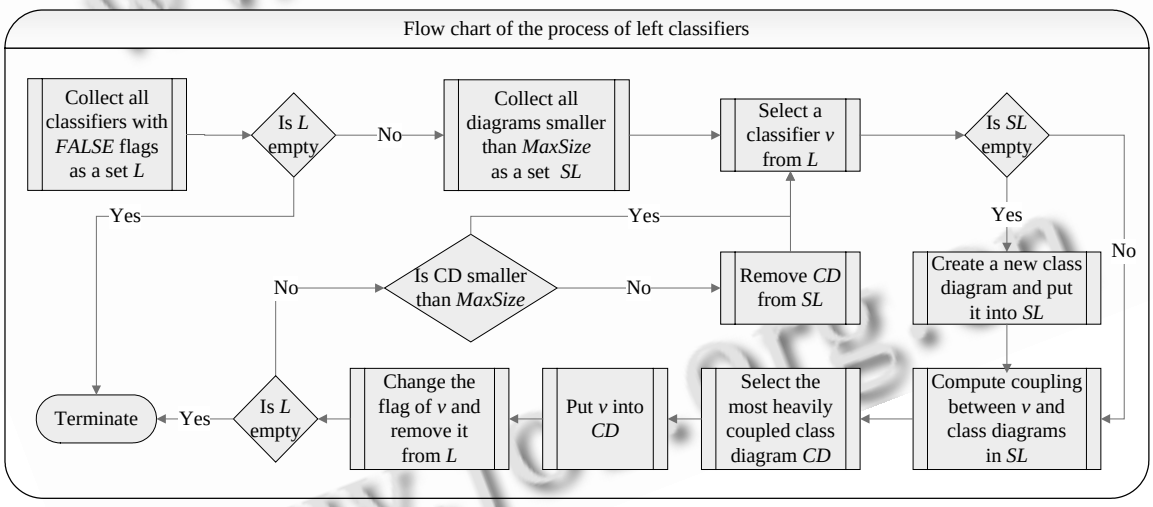

Fig.4 Flow chart of the process of left classifiers

图 4 零散类目处理流程示意图

依照 CBO 从大到小的顺序依次对每个类目(类或接口)v_{ur} 进行如下迭代处理(迭代流程如图 3 所示):

(1) 创建新类图.创建新类图 S={},{}表示 S 为不包含任何元素的空类图.

(2) 放入当前类目.将当前处理的类(或接口)v_{cur} 加入类图 S,S={v_{cur}}.

(3) 扩展类图.选出与类图 S 耦合关系最强的不属于类图 S 的类目 $v_i \notin S$.类目 v_i 和类图 S 的耦合度为 v_i 与 S 内所有类目的耦合度的总和.它反映了类目 v_i 与类图中所有类目的紧密程度.如果类图 S 与 v_i 的耦合度

大于 0 而且类图包含的类目个数不超过 *AverageSize*(|*S*|<*AverageSize*),则将 v_i 放入类图 *S*(*S*=*S*∪{v_i}),返回(3)继续选择下一个强耦合类.如此循环,直到类图大小达到 *AverageSize* 或者没有与类图耦合度大于 0 的类目为止.

(4) 如果类图包含的类目个数小于 *MinimalSize*(|*S*|<*MinimalSize*),则撤销该类图,对类目 v_{cur} 的迭代处理到此结束.太小的类图价值有限,不但增加了类图的数量,而且过细的切分可能导致难以全面理解和把握软件系统.

(5) 如果类图 *S* 中有 2/3 以上的类目已经出现在先前创建的小类图中(即“已分配”标志位为“true”),则撤销类图 *S*,对类目 v_{cur} 的迭代处理到此结束.类图间的大量冗余可导致类图数量过多.

(6) 如果类图 *S* 与某个先前创建的小类图 *S*′重叠部分超过 *S* 或者 *S*′的一半,即 $|S \cap S'| > \mathrm{Min}\{|S|,|S'|\}$,则合并 *S* 和 *S*′.如果合并后的类图大小超过 *MaxSize*,则需要剥离耦合度较低的类目,使得合并后的类图大小不超过 *MaxSize*.

(7) 对类图 *S* 内的所有类目,将其“已分配”标志位置为真“true”,表示这些类目已经至少出现在 1 个新类图中.

当所有类目依次迭代结束之后,部分迭代成功创建新类图,但也可能有部分迭代创建失败.所以,最后可能有部分与其他类目耦合度较低的类目没有出现在任何新创建的类图内(“已分配”标志位依然为假“false”).为了完整地拆分类图,需要将这些零散的类目归入某个类图,以防止类图拆分中的信息丢失.

对所有“已分配”标志位为假的类目 v_{left} 执行如下操作:

(1) 找出与该类目耦合度最大的类图 *S*.类目 v_{left} 和类图 *S* 的耦合度为

$$Couple(v_{left},S)=\sum_{v\in S}Couple(v_{left},v).$$

(2) 如果类图 *S* 的大小已经达到 *MaxSize*,则将类目 v_{left} 归入与 v_{left} 耦合强度仅次于 *S* 的类图 *S*′.以此类推,直到归入某个小于 *MaxSize* 的类图为止.如果所有类图大小均已达到最大值,则创建新类图以容纳零散类目.

2 实验验证

2.1 算法评价

算法评测的主要评测指标为算法的复杂度和有效性.算法复杂度包括时间复杂度和空间复杂度.

每个类目需要进行一轮迭代,而每轮迭代需要扩展最多 *AverageSize* 个类目,所以迭代的时间复杂度为 $O(n)$.而迭代结束后可能需要处理剩余的零散类目.处理时需要统计该类目与现有类图的耦合强度,统计的复杂度为 $O(n')$,其中,n'为已经拆分到新类图的类目数.所以,处理第 1 个零散类目的复杂度为 $O(n')$,处理第 2 个零散类目的复杂度的复杂度为 $O(n'+1)$,而处理最后一个的复杂度为 $O(n-1)$.所以,处理零散类目阶段的复杂度合计为 $O\left(\sum_{i=0}^{n-n'-1}(n'+i)\right)$.在实际系统中,通过迭代拆分到新类图中的类目要远远多于最后留下来的零散类目,即 n'接近于 *n*.所以,最后处理零散类目的时间远小于迭代处理的时间.因此,整个算法的实际时间复杂度约为 $O(n)$.

算法的空间需求主要用于存放大类图中的每个类目及其对应的“已分配”标志位($O(n)$)、类目间的耦合强度($O(n^2)$)等,所以,其空间复杂度为 $O(n^2)$.

算法的有效性较难衡量.要通过比较算法自动拆分的结果与手工拆分的结果之间的差异性来评价大类图拆分算法的有效性.但这种差异较难量化.评价算法有效性的另一种方法是通过比较直接手工划分类图与在自动划分的基础上调整类图的工作量.如果在自动划分的类图上的调整更为简单、容易,则说明自动拆分类图是有益的.因为工作量较容易度量和量化,所以通过工作量来比较相对较为容易,也较为客观.

2.2 实验目的

实验的目的是要揭示大类图自动拆分是否有助于调整和优化逆向工程生成的类图.通过实验需要回答如下几个问题:

(1) 与完全的手工拆分相比,在自动拆分的类图上进行优化和调整能否减少工作负担?如果能,那么能减少多少工作量?

(2) 自动拆分并手工优化得到的类图与完全手工拆分得到的类图在可读性和可理解性上有没有差异?有多大差异?

(3) 算法的时间复杂度是否如算法评测中分析的那样,为 $O(n)$?算法的时间复杂度是否可以接受?

第 1 个问题专注于自动拆分方法的实用价值:对优化逆向工程得到的类图是否有帮助?第 2 个问题专注于自动拆分方法的效果:得到的类图是否具有较高的质量?第 3 个问题关注于自动拆分的代价:时间复杂度是否可以接受?这 3 个问题较为全面地衡量了自动拆分方法的有效性和实用性.

2.3 实验对象

元建模系统 Meta-Modeler[9]是北京大学信息科学技术学院软件研究所开发的基于 MOF(meta-object facility,元对象设施)[15]的建模工具生成系统.元建模系统 Meta-Modeler 是一个大型的面向对象系统,耗时将近 30 人月(man-month),包含约 500 个类,4 万行 Java 代码.在这样一个大型的实际项目上实验验证算法具有较高的可信度.

项目开发人员主要以学生为主,较难执行严格的过程管理.多数学生只进行简单设计即进入代码编写,而且大部分简单设计也未能完整地保留下来.这导致项目设计文档残缺不全,不利于项目的后期维护与扩展.

为了弥补开发过程中设计文档缺失的不足,也为了提高软件系统的可理解性和可维护性,项目组试图通过逆向工程生成部分设计文档.同时,通过手工优化自动生成的设计文档来提高设计文档的质量,满足后续维护与扩展的需要.

2.4 实验过程

为了拆分类图,首先必须通过逆向工程工具从源代码生成类图.生成类图的组织方式目前主要有 3 种:比较简单的一种是将项目的所有类和接口放入一个大类图中;第 2 种方式是按照类和接口的命名空间(name space)来组织,每个命名空间对应一个类图;第 3 种方式是按包或目录组织,每个包或目录对应一个类图.第 1 种方式可能导致类图过大,而后两种方式的类图则不仅可能导致类图过大,还有可能因为划分过细而丢失关键信息.这是因为在后两种组织方式中,不同命名空间(包或目录)内的相关类的紧密耦合关系无法体现在类图中.

本文提出的大类图拆分方法意在拆分类图,而不是将分散在不同类图中的相关类合并到某个类图上,所以本实验选择第 1 种组织方式(项目内的所有类均放入一个大类图中),以充分展示类(接口)之间的耦合关系.

首先利用本文提出的大类图拆分方法对逆向工程生成的类图进行拆分.在自动拆分的基础上,可以根据需要手工调整和优化,以提高模型的质量.手工调整和优化的另一个目的在于对自动拆分方法的评估.调整的幅度、工作量等有助于衡量自动拆分算法的有效性.

为了对拆分效果进行更深入的评估,我们也要求开发人员对逆向工程生成的大类图进行手工拆分,创建一系列大小合适的类图,以展现系统的静态结构.通过比较完全手工拆分和在自动拆分的基础上进行调整这两种方法,可以有效地衡量自动拆分方法在优化逆向工程生成的软件模型上的作用.

实验目的中的第 1 个和第 3 个问题都可以通过量化度量(工作量及算法运行时间)来评价,而第 2 个问题的量化度量则较为困难.目前尚没有广为接受的类图可理解性和可维护性度量指标.因此,本实验将以专家评价的方式来比较类图的质量.由一组中立的专家独立地比较两种方式得到的类图,每位专家从中选出一个他认为具有更好的可读性和可理解性的类图.最后,通过统计每个类图得到的专家票数来衡量它们的质量.

2.5 实验结果

在“7±2”原则[10]的基础上适度放宽,本实验将类图的平均大小、最大值及最小值分别设置为:*AverageSize*=8, *MinSize*=4,*MaxSize*=14.实验结果见表 1.

Table 1 Experimental results

表 1 实验结果

		Semi-Automatical	Manual
Effort (man day)		1.5	6
Amount of class diagrams		87	92
Size of class diagrams	Minimal size	4	4
	Average size	8.2	7.4
	Maximal size	14	15
Amount of classes		487	487
Votes from experts		6	4

实验结果表明,在自动拆分的基础上进行优化可以大量节约手工调整和优化的时间.在本次实验中,自动拆分方法节约了 4.5 人天的工作量,约为全手工拆分工作总量(6 人天)的 75%,效果明显.

在拆分的质量上,两种方法所得的结果没有本质性差异.专家评价显示它们的可读性和可理解性相当(6 比 4).类图大小的度量结果也非常接近.类目总数都是 487 个,与系统包含的类目相当,表明两种拆分方法都没有丢失类目.

本实验的实验环境具体为:联想 ThinkPad T61 7664MU1,Intel Core2 Duo(1800MHz),2GB 内存,Microsoft Windows Vista Business (Service Pack 1).算法运行时间:221 秒(约 4 分钟).相比于手工拆分的 6 人天(man-day),4 分钟的运行时间是可以接受的.

3 讨 论

拆分方法允许同一个类目出现在多个类图中,以便展现该类目与其他类目间的关系.系统的核心类目通常与大量的类目交互,并在不同的交互中展现不同的角色.如果一个类目只允许出现在一个类图中,则该类目与其他元素之间的耦合关系将无法得到完整展现(信息丢失).另一方面,为了抑制不同类图间的信息冗余,拆分方法撤销严重重叠的类图以保证较低的信息冗余.

本文提出的大类图拆分方法需要确定新类图的期望大小,包括 3 个参数:平均大小 *AverageSize*、最小值 *MinimalSize* 以及最大值 *MaxSize*.但是不同的开发和维护人员可能难以在这 3 个参数上达成一致.不同的人有不同的阅读能力和阅读习惯,所以也必然导致他们对这 3 个参数的不同看法.幸运的是,Miller[10]的研究表明大多数人能够同时理解的事物都在一个大体相同的范围内(5~9).在实施自动拆分时,可以根据实际情况对这 3 个参数进行微调,以适应维护人员在阅读习惯和阅读能力上的差异.

在实验中,对两种方式(自动拆分后手工优化和全手工拆分)的工作量的比较需要考虑人的因素.因为人的个体差异,即使是完成相同的任务也可能需要不同的工作量(时间).为了减小人的因素对实验结果的干扰,使实验中的工作量对比具有更高的可信度,我们将参与实验的 20 个学生根据其能力和对实验对象的熟悉程度进行分组,力图均衡两个独立小组的能力和熟悉程度.

4 相关工作

逆向工程,尤其是以 UML 为基础的逆向工程已经取得了一定的进展,得到了广泛的应用.其中 UML 类图的逆向工程最为成熟,应用也最为广泛[7].现有的主流 UML 建模工具都提供了 UML 类图逆向生成的功能[2−4].每个工具具体的功能及性能比较参见文献[7]及相关产品的用户手册.

在类图的拆分及选择性展现方面,Kollmann 和 Gogolla[16]提出了一种基于度量的类图选择性表示方法.该方法首先通过度量指标选出类图中的若干中心点,然后从每个中心点开始扩展成一个新类图.该方法的主要问

题在于:

(1) 没有对新类图的大小进行控制.可能导致新类图的大小依然超过人的理解和把握极限.所以,该方法不够彻底.

(2) 没有处理零散类目.只是从若干中心点开始扩展,耦合较为松散的类目可能不会出现在任何新类图中,导致信息丢失.

本文提出的拆分方法从每个类目出发开始迭代,每次迭代将新类图的大小控制在预先设定的范围内,从而使得新类图的大小适合开发维护人员阅读和理解.其次,本文提出的拆分方法对所有未出现在新类图中的零散类目进行处理,将其归入关系最紧密的类图,保证了信息的完整性.

5 结论和进一步的工作

分析和设计模型在软件开发和维护中的重要性得到了广泛的认可,但是仍然有许多软件系统缺乏必要的分析和设计模型,加大了系统后续维护和扩展的难度.为了解决这个问题,人们利用逆向工程工具从源代码自动生成部分或全部系统模型.在逆向工程中,目前应用最广的是类图的逆向生成.但是,自动生成的类图通常过于庞大,缺乏必要的划分,影响了模型的可读性和可理解性.

本文针对这个情况提出了一种基于耦合度的大类图拆分方法.将逆向工程生成的大类图进行拆分,得到若干大小合适的新类图.大类图的划分提高了模型的可读性,同时保证了信息的完整性.通过在大型实际系统中的应用,本文提出的类图拆分方法的有效性得到验证.自动拆分方法能够大幅度地减少手工拆分的工作负担(减少幅度达 75%).

进一步的研究将继续提高拆分的有效性,降低手工调整和优化的工作负担.采用机器学习等方法理解和记忆人工拆分的经验,可能有助于提高拆分方法的效果,也有助于适应不同人群在阅读习惯和阅读能力上的差异.

References:

[1] Chikofsky EJ, Cross II JH. Reverse engineering and design recovery: A taxonomy. IEEE Software, 1999,7(1):13–17.

[2] http://www-306.ibm.com/software/awdtools/developer/rose/index.html

[3] http://www.borland.com/us/products/together/index.html

[4] http://jude.change-vision.com/jude-web/product/community.html

[5] Ma ZY, Zhao JF, Meng XW, Zhang WJ. Research and implementation of Jade Bird object-oriented software modeling. Journal of Software, 2003,14(1):97−102 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/97.htm

[6] OMG. UML2 Superstructure. OMG Document, formal/05-07-04, 2005.

[7] Kollmann R, Selonen P, Stroulia E, Systa T, Zündorf A. A study on the current state of the art in tool-supported UML-based static reverse engineering. In: Proc. of the 9th Working Conf. on Reverse Engineering (WCRE 2002). 2002.

[8] Shao WZ, Yang FQ. Object-Oriented Systems Analyze. 2nd ed., Beijing: Tsinghua University Press, 2006 (in Chinese).

[9] Ma ZY, Liu H, He X, Zhang L, Ji Z, Ge M. Model driven development. Acta Electronica Sinica, 2008,36(4):731−736 (in Chinese with English abstract).

[10] Miller GA. The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychological Review, 1956,63(2):8–97.

[11] Briand LC, Daly JW, Wüst JK. A unified framework for coupling measurement in object-oriented systems. IEEE Trans. on Software Engineering, 1999,25(1):91−121.

[12] Chidamber SR, Kemerer CF. A metrics suite for object oriented design. IEEE Trans. on Software Engineering, 1994,20(6): 476−493.

[13] Chidamber SR, Kemerer CF. Towards a metrics suite for object oriented design. SIGPLAN Notices, 1991,26(11):197−211. [doi: 10.1109/32.295895]

[14] Li W, Henry S. Object-Oriented metrics that predict maintainability. Journal of Systems and Software, 1993,23(2):111−122. [doi: 10.1016/0164-1212(93)90077-B]

[15] Object Management Group. Meta Object Facility (MOF) 2.0 Core Specification. OMG Adopted Specification ptc/04-10-15, 2004.

[16] Kollmann R, Gogolla M. Metric-Based selective representation of UML diagrams. In: Proc. of the 6th European Conf. on Software Maintenance and Reengineering (CSMR 2002). 2002. 1–10.

附中文参考文献:

[5] 麻志毅,赵俊峰,孟祥文,张文娟.青鸟面向对象软件建模工具的研究与实现.软件学报,2003,14(1):97−102. http://www.jos.org.cn/1000-9825/14/97.htm

[8] 邵维忠,杨芙清.面向对象的系统分析.第 2 版,北京:清华大学出版社,2006.

[9] 麻志毅,刘辉,何啸,张乐,吉喆,戈牧.一个支持模型驱动开发的元建模平台的研制.电子学报,2008,36(4):731−736.

刘辉(1978－),男,福建长汀人,博士,讲师,CCF 会员,主要研究领域为软件重构,面向对象建模,元建模,MDA.

邵维忠(1946－),男,教授,博士生导师,主要研究领域为软件工程,软件工程环境,面向对象技术,软件复用,构件技术.

麻志毅(1963－),男,博士,副教授,主要研究领域为软件工程,软件工程环境,面向对象技术.